IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A method including

simulating a plurality of dynamically-allocated threads using a single statically-allocated thread; and

maintaining state information regarding each dynamically-allocated thread maintained within said statically-allocated thread.

2. (Previously Presented) A method as in claim 1, further including maintaining, for a routine capable of being suspended or interrupted, a set of entry points into which said routine is capable of being re-entered after said suspension or interruption.

3. (Previously Presented) A method as in claim 1, further including generating said set of entry points in response to one or more programming macros.

4. (Previously Presented) A method as in claim 1, further including maintaining high concurrency among threads without maintaining a substantial amount of state information regarding simulated threads.

5. (Previously Presented) A method as in claim 1, wherein said state information includes a relatively small procedure call stack for the simulated threads.

6. (Previously Presented) A method as in claim 1, wherein said state information includes a relatively small collection of local variables and other state information for the simulated threads.

7. (Currently Amended) Apparatus including a file server system having a single statically-allocated thread including a plurality of simulated dynamically-allocated threads, said statically-allocated thread including state information regarding each said simulated thread.

8. (Previously Presented) Apparatus as in claim 7, further including a routine capable of being suspended or interrupted, said routing having a set of entry points into which said routine is capable of being re-entered after said suspension or interruption.

9. (Original) Apparatus as in claim 8, wherein said set of entry points are responsive to one or more programming macros.

10. (Previously Presented) Apparatus as in claim 7, wherein said state information includes a relatively small procedure call stack for the simulated threads.

11. (Previously Presented) Apparatus as in claim 7, wherein said state information includes a relatively small collection of local variables and other state information for the simulated threads.

12. (Previously Presented) A method as in claim 1, wherein said plurality of dynamically-allocated threads are simulated using said statically-allocated thread under an operating system that is incapable of executing plural actual dynamically-allocated threads.

13. (Previously Presented) A method as in claim 1, wherein said statically-allocated thread simulates said plurality of dynamically-allocated threads by using a scheduler to call thread blocks for said plurality of dynamically-allocated threads.

14. (Previously Presented) A method as in claim 13, wherein said thread blocks are stored in a linked list maintained by said statically-allocated thread.

15. (Previously Presented) A method as in claim 14, wherein said thread blocks in said linked list are called in turn by said scheduler.

16. (Previously Presented) A method as in claim 4, wherein an amount of state information that is maintained is less than an amount of state information that would be necessary for plural actual dynamically-allocated threads.

17. (Previously Presented) A method as in claim 5, wherein said relatively small procedure call stack is smaller than a procedure call stack that would be necessary for plural actual dynamically-allocated threads.

18. (Previously Presented) Apparatus as in claim 7, wherein said file server system is incapable of executing plural actual dynamically-allocated threads.

19. (Previously Presented) Apparatus as in claim 7, where said statically-allocated thread simulates said plurality of dynamically-allocated threads by using a scheduler to call thread blocks for said plurality of dynamically-allocated threads.

20. (Previously Presented) Apparatus as in claim 19, wherein said thread blocks are stored in a linked list maintained by said statically-allocated thread.

21. (Previously Presented) Apparatus as in claim 20, wherein said thread blocks in said linked list are called in turn by said scheduler.

22. (Previously Presented) Apparatus as in claim 10, wherein said relatively small procedure call stack is smaller than a procedure call stack that would be necessary for plural actual dynamically-allocated threads.

23. (Previously Presented) Apparatus as in claim 11, wherein said relatively small collection of local variables and other state information is smaller than a collection of local variables and other state information that would be necessary for plural actual dynamically-allocated threads.

24. (Currently Amended) A method of implementing a plurality of simulated dynamically-allocated threads using a single statically-allocated thread, comprising:

using a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads; and

maintaining state information regarding each of said plurality of simulated dynamically-allocated threads.

25. (Previously Presented) A method as in claim 24, wherein said thread blocks are stored in a linked list maintained by said statically-allocated thread.

26. (Previously Presented) A method as in claim 25, wherein said thread blocks in said linked list are called in turn by said scheduler.

27. (Currently Amended) Apparatus including a server that implements a plurality of simulated dynamically-allocated threads using a single statically-allocated thread, comprising:

a processor that executes a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads; and

memory that stores state information regarding each of said plurality of simulated dynamically-allocated threads.

28. (Previously Presented) Apparatus as in claim 27, wherein said thread blocks are stored in a linked list maintained in said memory by said statically-allocated thread.

29. (Previously Presented) Apparatus as in claim 28, wherein said thread blocks in said linked list are called in turn by said scheduler.